International Journal of Engineering Research (ISSN : 2319-6890)
Volume No.2, Issue No.3, pp : 225-228 01 July 2013

Concept of a Supervector Processor: A Vector Approach to Superscalar Processor, Design and Performance Analysis

Deepak Kumar, Ranjan Kumar Behera, K. S. Pandey

National Institute of Technology, Hamirpur (H.P)
deepakkumar.nith@gmail.com; jranjanb.19@gmail.com; kumar@nith.ac.in

ABSTRACT- ***To maximize the available performance is always a goal in microprocessor design. In this paper a new technique has been implemented which exploits the advantage of both superscalar and vector processing technique in a proposed processor called Supervector processor. Vector processor operates on array of data called vector and can greatly improve certain task such as numerical simulation and tasks which requires huge number crunching. On other hand superscalar processor issues multiple instructions per cycle which can enhance the throughput. To implement parallelism multiple vector instructions were issued and executed per cycle in superscalar fashion. Case study has been done on various benchmarks to compare the performance of proposed supervector processor architecture with superscalar and vector processor architecture. Trimaran Framework has been used in order to evaluate the performance of the proposed supervector processor scheme***.

KEYWORDS- Supervector processor, Superscalar processor, SUIF, Trimaran, Vector processor.

I. INTRODUCTION

The performance of a computing system can be determined by three factors, first the number of processor instruction required to execute a program, second processor cycle time, and lastly the average number of processor cycles required to execute an instruction[1]. The performance can be improved by reducing any of the three factors. Generally cycle time can be reduced by different implementation technology, number of instructions by optimizing compilers and average number of cycle per instruction can be reduced through different processor architecture [I]. Here we have tried to improve the performance by reducing the average number of processor cycle required to execute a program by introducing proposed supervector processor architecture. The simplest processor is the scalar processor which executes one instruction per cycle that means only one instruction is issued per cycle and only one instruction is expected to complete from the pipeline per cycle.

Superscalar processors are able to execute more than one instruction in a clock cycle by exploiting instruction level parallelism. It consists on number of pipelines that are working in parallel. Depending on the number and kind of various parallel functional units available the processor can executes certain number of instructions at a time [II]. Thus the advantage of superscalar processor is it can greatly improve the performance by increasing throughput since multiple instructions can be issued per cycle and multiple results can be generated per cycle. A study [I] concludes that a superscalar processor can have nearly twice the performance of a scalar processor but it requires four major hardware features: branch prediction, out of order execution, register renaming, and a four instruction decoder. These features are independent and removing any one can degrade the performance by 18 percent.

Vector Processors are similar to scalar processor but differ only in that it performs calculation on a vector as a whole. The operands to the instructions are complete vectors instead of one element. Thus vector processors reduce the fetch and decode bandwidth as the number of instructions fetch are less [3], hence contributing to performance improvement. It also exploits data level parallelism in large scientific and multimedia application.

In the proposed supervector architecture we have tried to implement a processor architecture which takes advantage of both superscalar as well as vector processing technique by issue and execution of multiple vector instruction in superscalar fashion. The performance of the proposed processor has been compared and has been shown graphically.

The remainder of this paper is organized as follows: Section II briefly discusses related work on the improvement of superscalar processors. In section III we have explained the architecture of our proposed supervector architecture with a block diagram and various components of the proposed processor. The various tools that has been used for designing and performance evaluation of supervector processor has been briefly discussed in section IV.

The experimental setup and result has been presented in section V followed by conclusion in section VI.

II. RELATED WORK

Several works has been done previously in order to improve the performance of superscalar processors. In [II] a new algorithm had been proposed for superscalar processors based on changing Branch Target Buffer structure to reduce the misprediction penalty. Another previous work [III] shows that to fully exploit superscalar processor of degree n, n instruction must be executed in parallel all the time which can be restricted by if stall and dead time occurs thus reducing the performance and it focused on the importance of software schedulers that can improve the performance of the compiled code to best utilize the inherent property of superscalar processor. One of the works [IV] has proposed an adaptive interleaved multi threading technique which is a proven technique to improve the efficiency of vector resources so to provide gain in speed as well as reduction in consumption of energy.

In this work we have tried to improve the performance of superscalar processor by introducing vector operands in place of scalar operands with the help of vectorization technique in proposed supervector processor so that it can take the advantage of both superscalar and vector processing technique.

III. SUPERVECTOR PROCESSOR ARCHITECTURE

Supervector processor takes advantage of both superscalar processor as well as vector processor by exploiting both instruction level parallelism as well as data level parallelism simultaneously. Instruction scheduling in supervector processor is dynamic i.e. it is done at runtime by hardware. Supervector processor uses vector instruction set which is a compact way to encode large amount of data in an array .Supervector processor consists of vector register file which is used as intermediate storage and also provides interconnection to functional units. It also contains of scalar register for those instructions that contain scalar operands. Multiple redundant vector functional units (FU) are used that works in parallel in superscalar fashion to increase throughput. It has a vector load store unit which is used for loading or storing vector to or from memory. Fig 1 shows the architecture of proposed supervector processor. Supervector processor fetches and decodes vector instructions several instruction at a time. These vector instructions are analyzed for dependencies. Independent vector instructions are issued for parallel, execution primarily based on availability of vector operands known as dynamic scheduling. After completion it is reordered before commit in order to maintain original program sequence.

IV. FRAMEWORK USED FOR PERFORMANCE EVALUATION

Trimaran has been used as a framework in order to design and evaluate the performance of supervector processors [V]. It provides an integrated compiler and simulation infrastructure for research in computer architecture and compiler optimizations. It is highly parameterizable and can target wide range of architecture. It supports different ISA like HPL-PD which is a parametric architecture which can be configured to admit Superscalar architecture implementation. The proposed supervector architecture includes combination of superscalar architecture implementation along with vector processing technique. Trimaran also has one of its first kinds of back-end vectorizer which can be used to extract and exploit data level parallelism using short vector instruction.

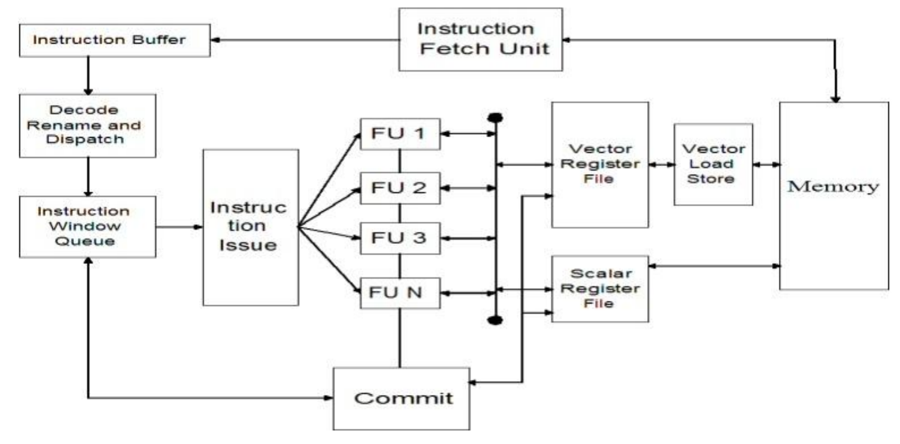


Fig 1: Supervector Architecture

Fig 2 shows the various components and compilation steps involved in Trimaran. It consists of three components: OpenIMPACT compiler, Elcor compiler and Simu simulator.

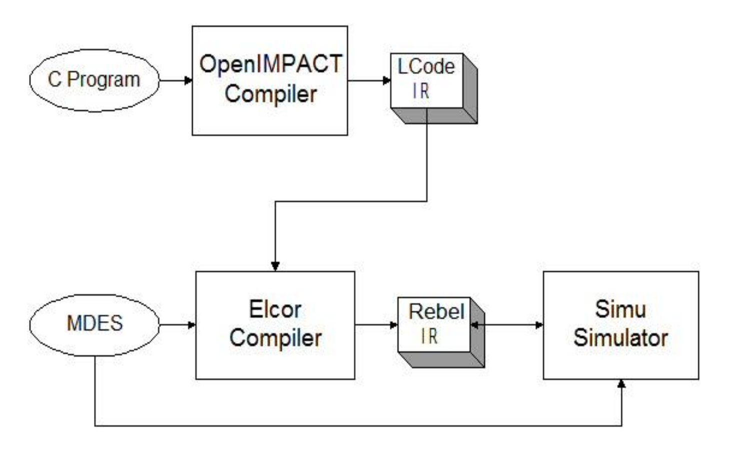


Fig 2: Trimaran System Infrastructure

Trimaran uses OpenIMPACT compiler to compile the original source code into assembly intermediate representation Lcode. Lcode is optimized for ILP but not for specific machine, it again passes through Elcor Compiler along with machine description (MDES) that specifies the target machine. We have modified the machine description (MDES) in order to support target processor architecture i.e. superscalar and supervector processor architecture as per our requirement. Elcor compiles the code as per target architecture mentioned in MDES file producing another intermediate code REBEL. The component SIMU is the simulator of Trimaran which consumes the REBEL code and executes it to generate various execution statistics. To implement supervector processing technique, scalar code has been vectorized in order to identify and exploit data level parallelism for supervector processor, for this back end-vectorizer provided by Trimaran has been used , but it is most effective and in many case only applicable when it has precise memory dependency information. For this we have used SUIF as a front end to extract dependence information. SUIF is an extensive compiler system that supports parallelization. It also provides free infrastructure to support collaborative research in optimizing and parallelizing compilers.

V. EXPERIMENTAL RESULTS

Vectorization of the program has been done which is a process of converting a computer program from a scalar implementation, which processes a single pair of operand at a time to a vector implementation which processes one operation on multiple pairs of operands at once. The concept of vectorization is very important for supervector processors since vector instructions will be executed in superscalar fashion to realize this architecture. Simulation result with and without vectorization has been shown in TABLE I and Fig 3.

TABLE I: Total cycles taken with and without vectorization

Benchmark	Total Cycles (without vectorization)	Total Cycles (with vectorization)	Increase in Performance
Sqrt	2982	1257	57.84 %
Hyper	3518	3348	4.83 %
Ifthen	4720	4307	8.75 %
Strcpy	15394	11832	23.13 %

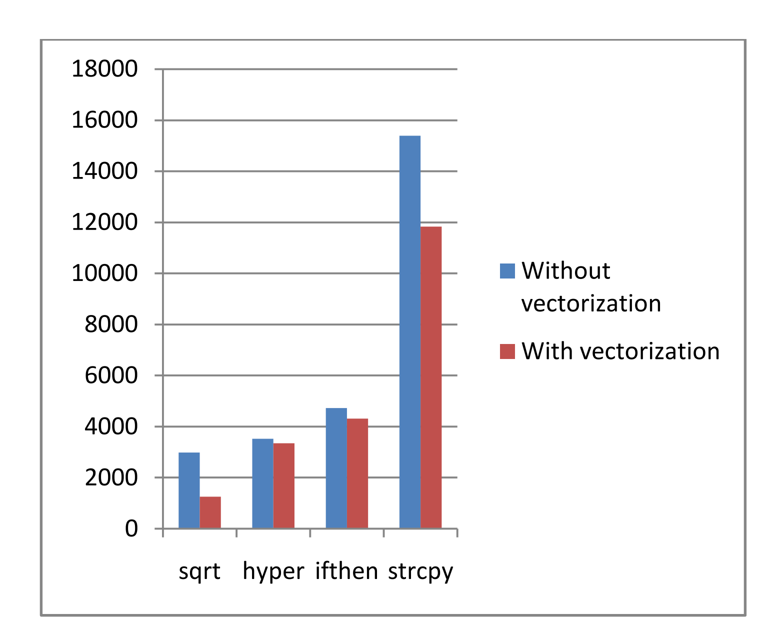


Fig 3: comparison of total cycles with and without vectorization

Table I shows the total cycles required to execute the benchmarks with and without vectorization of operands. Fig 3 compares the execution statistics showing the importance of vectors w.r.t to scalars. Since supervector processor takes advantage of both superscalar and vector processor technique, the target supervector processor has been configured by modifying the parameters present in machine description of Trimaran in such a way that it was able to issue multiple instructions in one clock cycle. In this case it is configured to issue 4 instructions in one clock cycle. Also many parameters to support vector processing were configured accordingly. Vector

length was kept as 4. We have used SUIF in order to generate precise memory dependence information so that vectorization can be done efficiently. Simulation has been done on various benchmarks present in Trimaran directory. First they have been simulated for superscalar processor, vector processor and then for the designed supervector processor. Table II and Fig 4 depict the simulation result.

TABLE II: Total cycles comparison for different benchmarks

Bench mark	Total Cycles (Superscalar)	Total Cycles (Vector)	Total Cycles (Supervector)	Percentage Increase in Performance w.r.t	
				Superscalar	vector
Sqrt	2741	1257	1167	57.4 %	7.15 %
Ifthen	4209	4307	3804	9.62 %	11.67 %
Hyper	3130	3348	2966	5.23 %	11.4 %
Strcpy	14438	11832	10568	26.8 %	10.68 %

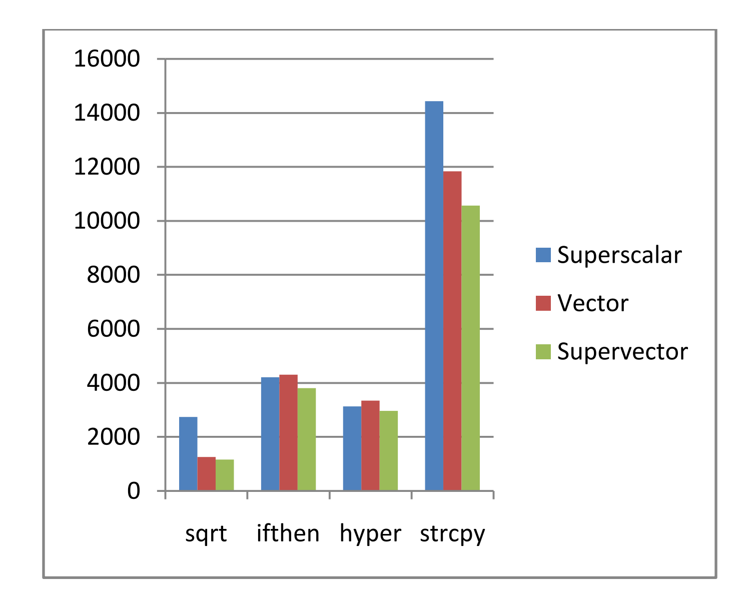


Fig 4: Total Cycle comparison for different benchmarks

Table II shows the total cycles required in order to execute the various benchmarks in superscalar, vector and supervector processor architecture. We can see that the total cycles required to execute benchmark is less in supervector processor architecture when compared with superscalar and vector processor. Thus there is a percentage increase in performance of the proposed supervector with respect to superscalar and vector processors. Also the comparison in Fig 4 shows that total cycles required by our designed supervector processor architecture to execute the benchmarks are less than superscalar and vector processor, hence it is clear that the performance of proposed supervector processor is better than vector and superscalar processor architecture.

VI. CONCLUSION

Supervector processor takes advantage of superscalar as well as vector processor and its performance is better than both. Enhanced parallelism has been incorporated by superscalar issue and execution of vector instructions. Since supervector processor operates on vector operands, it is very efficient in extracting data level parallelism and greatly improves performance in array intensive application. So supervector processor has a promising future in a field that requires large amount of numeric calculation and array intensive work.

REFERENCES

i. *William M. Johnson, "Super-scalar processor design", Technical report no CSL-TR-89-383, June 1989.*

ii. *Rubina Khanna, Sweta Verma, Dr. Ranjit Biswas, Prof J.B. Singh, "Implementation of branch delay in superscalar processors by reducing branch penalties", IEEE 2nd International Advance Computing Conference 2010.*

iii. *Glass, D.N. "Compile time instruction scheduling for superscalar processors", 35 IEEE computer society International Conference 1990.*

iv. *Valeriu Codreanu, Lucian Petrica, Radu Hobincu, "Increasing Vector Processor Pipeline Effeciency with a Thread-Interleaved Controller".*

v. *Trimaran 4.0 manual, http://trimaran.org/documentation.shtml*

vi. *James E Smith, Gurinder S Sohi, "The microarchitecture of Superscalar processors", Proceedings of IEEE 1995.*

vii. *Basil Sh. Mahmood, Mamoon A. Al Jbaar, "Design and Implementation of SIMD vector Processor on FPGA", Fourth International Symposium on Innovation in Information & communication Technology, 2011.*

viii. *Vinod Kathail, Michael S. Schlansker, B. Ramakrishna Rau, "HPL-PD Architecture Specification" , HPL-93-80 (R.1), February 2000*

ix. *J.Hanessey and D.Patterson, "Computer Architecture: A Quantitative approach", Morgan Kaufmann, San Francisco, CA, 3rd edition 2002. Page 172-81*

x. *Vector Processors, http://www.cs.umd.edu/class/fall2001/cmsc411/proj01/cache/vector.html*